KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Exact algorithms for the equitable traveling salesman problem

Kinable J, Smeulders B, Delcour E, Spieksma F.

KBI_1610

Exact algorithms for the Equitable Traveling Salesman Problem*

Joris Kinable[†] [‡] Bart Smeulders[§] Eline Delcour
Frits C.R. Spieksma[§]

March 31, 2016

Abstract

Given a weighted graph $G = (V, E)$, the Equitable Traveling Salesman Problem (ETSP) asks for two perfect matchings in G such that (1) the two matchings together form a Hamiltonian cycle in G and (2) the absolute difference in costs between the two matchings is minimized. The problem is shown to be NP-Hard, even when the graph G is complete. We present two integer programming models to solve the ETSP problem. One model is solved through branch-and-bound-and-cut, whereas the other model is solved through a branch-and-price-and-cut framework. A simple local search heuristic is also implemented. We conduct computational experiments on different types of instances, often derived from the TSPLib. It turns out that the behavior of the different approaches varies with the type of instances; however, the branch-and-bound-and-cut approach implemented in Cplex seems to work best overall.

1 Introduction

We consider the following variation of the Traveling Salesman Problem (TSP). Given is an edge-weighted graph $G = (V, E)$, with $|V|$ even, and with edge-costs d_e for each $e \in E$. The cost of a matching in G is defined as the sum of edge-costs of the edges in the matching. The problem is to find two perfect matchings in G such that (i) the two matchings form a Hamiltonian cycle in G, and (ii) the absolute difference of the costs of these two matchings is minimum. Notice that a feasible solution need not exist. We call this problem the Equitable Traveling Salesman Problem, or ETSP for short.

This name is motivated by the following, more frivolous, description of our problem: two friends, in possession of a single bike, have agreed to jointly visit all given cities, i.e., to construct a tour. In addition, they have agreed to use the bike as follows: one friend rides (pedals) the bike, while the other sits on the bike's back. Directly after having visited a city, the two friends interchange roles. The objective in this problem is to find a tour such that the difference between the distances pedalled by each of the two friends, is minimum. An example of an instance of the ETSP is given in Figure 1.

We see the ETSP as an example of a class of combinatorial optimization problems referred to as *balanced* optimization problems. Balanced optimization problems differ form regular optimization problems in the sense that, informally speaking, costs of different "parts" of the solution should be close to each other; this happens in situations where an equal or fair distribution of resources/costs is pursued. One consequence is that 0 is a lowerbound for any optimum value (which is not necessarily true in a regular optimization problem). The general mathematical form of balanced optimization problems can be stated as:

$$\underset{S \in \mathcal{F}}{\text{minimize}} \{\max_{s \in S} w(s) - \min_{s \in S} w(s)\} \tag{1}$$

where $\mathcal{F}$ is a family of feasible subsets (solutions) of some superset $\mathscr{S}$, and $w(s)$ a cost (weight) function which assesses the cost (weight) of element $s \in \mathscr{S}$. For instance, in the context of ETSP, superset $\mathscr{S}$ encompasses all possible perfect matchings in the graph G, and $\mathcal{F}$ comprises of all subsets $S \subseteq \mathscr{S}$ such

*This research has been partially funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office; the research was carried out when the first author was at KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC - Belgium.

[†]Robotics Institute & Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave Pittsburgh, PA 15213, USA Email: {jkinable@cs.cmu.edu}

[‡]Former: KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC - Belgium

[§]KU Leuven, Faculty of Business and Economics, ORSTAT, Belgium. Email: {bart.smeulders;frits.spieksma}@kuleuven.be.

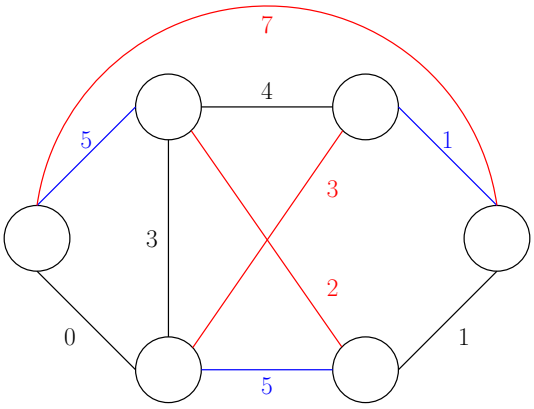


Figure 1: Example of an ETSP instance and a solution. The perfect matching consisting of the blue edges has a cost of 12 and the perfect matching consisting of the red edges has a cost of 11. The vale of the resulting solution equals 1.

that $|S| = 2$ and the two matchings in S form a Hamiltonian cycle. The objective function in (1) minimizes the imbalance: the absolute difference in costs between the most expensive and least expensive element in the solution.

A general class of balanced optimization problems as defined by equation (1) has been characterized first by Martello et al. (1984). Their focus is on balanced optimization problems that are solvable in polynomial time. Many optimization problems identified in literature can be reduced to the aforementioned structure. A number of examples can be found in Zeitlin (1981); Martello et al. (1984); Camerini et al. (1986); Katoh and Iwano (1994); Berežný and Lacko (2005); Cappanera and Scutellà (2005); Becker (2010); Delcour (2012); Ficker et al. (2016) - most of these results either involve polynomial-time algorithms for specific balanced optimization problems, or heuristics for balanced optimization problems that are (NP-)hard. In this paper, we describe exact approaches for a particular balanced optimization problem that is NP-hard; we see the outcome of this study as a first step towards understanding the behavior of different types of approaches that can be applied to NP-hard balanced optimization problems.

A problem that is closely related to (but different from) the ETSP is the so-called *balanced* TSP, studied by Larusic and Punnen (2011). In the balanced TSP, one seeks to minimize the difference between the largest edge-cost and the smallest edge-cost of edges used in a tour. Larusic and Punnen (2011) develop several heuristics, mainly relying on lower and upper bounding procedures, to solve the balanced TSP. They mention that their algorithms can be used to solve an optimization problem originating in the maintenance of aircraft engines. It is not difficult to define a problem that is a generalization of both the ETSP and the balanced TSP. Indeed, given an edge-weighted graph G, and an integer k, consider the question: do there exist k disjoint matchings $M_1, M_2, \ldots, M_k$ with $M_i \subset E, |M_i| = \frac{|V|}{k}$ for $i = 1, \ldots, k$ such that $\cup_i M_i$ forms a Hamiltonian cycle in G? Observe that for $k = 2$ the ETSP arises, while for $k = |V|$ the balanced TSP arises (see Kinable (2014)).

The following optimization problem is also related to the ETSP: Bassetto and Mason (2011) explore a periodic routing problem, where two tours of minimal total length through a number of nodes (customers) need to be found. Some customers must occur in both tours, others must occur in exactly one tour. The absolute difference between the number of customers in each tour is restricted from above, thereby obtaining two balanced tours. Notice that Bassetto and Mason (2011) do not consider reducing the imbalance between the two tours as part of their objective function; instead, the maximum allowed imbalance is part of the problem input and enforced through a constraint in the model.

Another problem, which structurally bears strong resemblance to the ETSP problem, is the Market-Split problem (Cornuéjols and Dawande, 1998; Aardal et al., 2000). The Market-Split problem attempts to minimize the total amount of slack (positive and negative) which has to be added to a set of diophantine equations to make the system feasible. The problem is often introduced with the example of a company having two sales divisions responsible for supplying retailers with products. The objective is to allocate each retailer to either one of the divisions such that each division controls a predefined fraction of the market for a given product; deviation of these predefined fractions should be minimized. A similar structure is present in the ETSP.

The goal of this work is to investigate how to solve instances of ETSP to optimality. For that purpose, we study and compare exact solution methods based on two integer programming formulations

for the ETSP. We will detail a branch-and-price approach for our problem, and compare it with a more traditional branch-and-bound method. In addition, we describe a local search method, and show how all these methods fare on different classes of instances of the ETSP.

The paper is organized as follows. Section 2 analyzes the complexity of the ETSP, showing that the problem, including some special cases, is NP-Hard. Section 3 introduces two integer programming formulations for the ETSP problem. A branch-and-price framework to solve one of these formulations is outlined in Section 4, and the local seacrh algorithm is described in Section 5. Outcomes of computational experiments for the ETSP are reported in Section 6. Section 7 offers the conclusion.

2 Complexity Analysis

In this section we analyze the computational complexity of the ETSP. We show that deciding whether a feasible solution exists to an instance of ETSP is NP-complete, and we show that, even for a complete graph, it is NP-hard to find an optimum solution. Let us formally state the decision version of ETSP:

Input: an undirected graph $G = (V, E)$, with $|V|$ even.

Goal: do there exist two disjoint perfect matchings M_1 and M_2 with $M_1 \subset E$ and $M_2 \subset E$ such that $M_1 \cup M_2$ is an Hamiltonian cycle in G?

It is not difficult to verify that the decision version of the ETSP is at least as hard as deciding whether a graph with an even number of nodes has a Hamiltonian cycle. Hence we can state the following theorem:

Theorem 1. *The decision version of ETSP is NP-complete.*

The optimization version of ETSP, simply referred to as ETSP, involves a cost d_e for each edge $e \in E$. Again, let us formally state the problem, where, for a subset of the edges $Q \subseteq E$, we use the following notation: $c(Q) = \sum_{e \in Q} d_e$.

Input: an undirected, weighted graph $G = (V, E)$, with edge costs d_e for all $e \in E$, and with $|V|$ even.

Goal: find two disjoint perfect matchings $M_1, M_2 \subset E$ such that $M_1 \cup M_2$ forms a Hamiltonian cycle in G, and such that $|c(M_1) - c(M_2)|$ is minimum.

Even for complete graphs, ETSP is a difficult problem:

Theorem 2. *ETSP is NP-hard for complete graphs.*

Proof. We use a reduction from Hamiltonicity: given an undirected graph $H = (W, F)$, does H contain a Hamiltonian cycle? We assume (wlog) that $|W|$ is even. We now build a complete, edge-weighted graph $G = (V, E)$ that forms the instance of ETSP. Let $V = W$, and for each edge $e \in F$, we introduce an edge $e \in E$, with edge cost $d_e = 0$. Consider now the edges not in F; we (arbitrarily) denote them by $\{e_1, e_2, \ldots, e_p\}$, with $p = |E \setminus F|$. Each of these edges is also present in E; we set $d_{e_j} = 2^j$, $j = 1, \ldots, p$. This completes the instance of ETSP.

We now argue that the existence of a Hamiltonian cycle is equivalent to the instance of ETSP having an optimum solution with value 0. Clearly, if there exists a Hamiltonian cycle in H then there is a unique decomposition of this cycle into two disjoint perfect matchings, each with cost 0. This leads to a solution of ETSP with value 0. On the other hand, suppose that the instance of ETSP has a solution with value 0. Thus, the difference between the costs of the two matchings forming a Hamiltonian cycle equals 0. Consider the most expensive edge in this pair of matchings, and denote its cost by d_{max}. If $d_{max} > 0$, then, by choice of the edge-costs, the sum of the edge-costs in the other matching will be less than d_{max}. Hence, no solution with value 0 exists. It follows that $d_{max} = 0$, which implies that all edges in the two matchings that form a Hamiltonian cycle have cost 0, leading to a Hamiltonian cycle in the graph H. □

We end this section by defining the concept of an *equitable* tour.

Definition 3. *A tour is called* equitable *if the two matchings composing the tour have equal cost, i.e., if the value of the objective function of an instance of ETSP equals 0.*

3 Formulations for the ETSP

In this section, we introduce two integer programming formulations of the ETSP. In fact, we will treat a slightly more general problem where the edge-sets, and the edge-costs, need not be the same for the two matchings. We use E_B and E_R to denote the two edge-sets; E_B refers to the 'blue' edges that can be used for one matching, while E_R refers to the 'red' edges to be used for the other matching. More precisely, we are given a graph $G = (V, E_B \cup E_R)$, and, let $\mathcal{M}_B$ ($\mathcal{M}_R$) refer to the set of perfect matchings in (V, E_B) ((V, E_R)). Each edge e in E_B (E_R) has a cost d_e^b (d_e^r). The cost of a matching $M \in \mathcal{M}_B$ is defined as $c^b(M) = \sum_{e \in M} d_e^b$. Analogously, the cost of a matching $M \in \mathcal{M}_R$ is $c^r(M) \equiv \sum_{e:e \in M} d_e^r$. Note that this problem definition does not require that $E_B \cap E_R = \emptyset$. Furthermore, a single edge $e \in E_B \cap E_R$ may have different weights in the red or the blue matching (i.e., $d_e^b = d_e^r$ does not necessarily hold). Finally, we define $\delta(S)$, $S \subseteq V$ as the set of edges having exactly one endpoint in S. Additionally, $\delta(v), v \in V$ is used as shorthand notation for $\delta(\{v\})$.

3.1 Formulation FBB

The first formulation uses a binary variable for each edge $e \in E_B$:

$$x_e^b := \begin{cases} 1 & \text{if edge } e \text{ is selected in the blue matching,} \\ 0 & \text{otherwise,} \end{cases}$$

as well as a binary variable for each edge $e \in E_R$:

$$x_e^r := \begin{cases} 1 & \text{if edge } e \text{ is selected in the red matching,} \\ 0 & \text{otherwise.} \end{cases}$$

$$FBB: \min \quad |\sum_{e \in E_B} d_e^b x_e^b - \sum_{e \in E_R} d_e^r x_e^r| \tag{2}$$

$$\text{s.t.} \quad \sum_{e \in \delta(v) \cap E_B} x_e^b = 1 \qquad \forall v \in V \tag{3}$$

$$\sum_{e \in \delta(v) \cap E_R} x_e^r = 1 \qquad \forall v \in V \tag{4}$$

$$x_e^b + x_e^r \leq 1 \qquad \forall e \in E_B \cap E_R \tag{5}$$

$$\sum_{e \in \delta(S) \cap E_B} x_e^b + \sum_{e \in \delta(S) \cap E_R} x_e^r \geq 2 \qquad \forall S \subset V, |S| \geq 3 \tag{6}$$

$$x_e^b \in \{0, 1\} \qquad \forall e \in E_B \tag{7}$$

$$x_e^r \in \{0, 1\} \qquad \forall e \in E_R \tag{8}$$

Strictly speaking, the formulation above is not linear due to the absolute value present in the objective function. A standard trick exists to make the formulation linear: using an additional variable, say w, adding two constraints of the form $w \geq \sum_{e \in E_B} d_e^b x_e^b - \sum_{e \in E_R} d_e^r x_e^r$ and $w \geq \sum_{e \in E_R} d_e^r x_e^r - \sum_{e \in E_B} d_e^b x_e^b$, and replacing the objective function by min w makes the formulation linear. For reasons of compactness we use formulation (2)-(8). Further, constraints (3) and (4) ensure that each vertex is incident to exactly one edge from the blue matching and one edge from the red matching. Constraints (5) imply that each edge can be used by at most one matching. Constraints (6) model the subtour elimination constraints (notice that there is an exponential number of them). Observe that model FBB remains a correct formulation of the ETSP when all subtour constraints with $|S|$ odd are removed from the formulation; this is a consequence of the fact that any integral subtour must have an even number of edges due to the alternation of blue and red edges. We chose to work with all subtour elimination constraints, since the linear programming relaxation of FBB is affected by the use of subtour elimination constraints for subsets with an odd number of nodes. Finally, constraints (7) and (8) are the integrality constraints.

When we speak of the solution of the linear relaxation of FBB (which we denote by LFBB), we refer to a solution (x_e^b, x_e^r) satisfying (3)-(6), $x_e^b, x_e^r \geq 0$, for which the value $|\sum_{e \in E_B} d_e^b x_e^b - \sum_{e \in E_R} d_e^r x_e^r|$ (which we denote by v_{BB}) is minimum.

3.2 Formulation FBP

Our second formulation has a variable for each perfect matching in the graph. More precisely, we define a binary variable for each perfect matching $M \in \mathcal{M}_B$ in (V, E_B):

$$z_M^b := \begin{cases} 1 & \text{if perfect matching } M \text{ is selected as the blue matching,} \\ 0 & \text{otherwise,} \end{cases}$$

as well as a binary variable for each perfect matching $M \in \mathcal{M}_R$ in (V, E_R):

$$z_M^r := \begin{cases} 1 & \text{if perfect matching } M \text{ is selected as the red matching,} \\ 0 & \text{otherwise.} \end{cases}$$

$$FBP: \min \quad |\sum_{M\in\mathcal{M}_B} c^b(M)z_M^b - \sum_{M\in\mathcal{M}_R} c^r(M)z_M^r| \tag{9}$$

$$\text{s.t.} \quad \sum_{M\in\mathcal{M}_B} z_M^b = 1 \tag{10}$$

$$\sum_{M\in\mathcal{M}_R} z_M^r = 1 \tag{11}$$

$$\sum_{M\in\mathcal{M}_B:\ e\in M} z_M^b + \sum_{M\in\mathcal{M}_R:\ e\in M} z_M^r \le 1 \qquad \forall e \in E_B \cap E_R \tag{12}$$

$$\sum_{e\in\delta(S)} \Big(\sum_{M\in\mathcal{M}_B: e\in M} z_M^b + \sum_{M\in\mathcal{M}_R: e\in M} z_M^r \Big) \ge 2 \qquad \forall S \subset V, |S| \ge 3 \tag{13}$$

$$z_M^b \in \{0,1\} \qquad \forall M \in \mathcal{M}_B \tag{14}$$

$$z_M^r \in \{0,1\} \qquad \forall M \in \mathcal{M}_R \tag{15}$$

The same comment we made with respect to the linearity of formulation FBB applies to formulation FBP as well. Formulation FBP selects two perfect matchings by constraints (10) and (11). Constraints (12) ensure that an edge in the intersection of E_B and E_R can be used by at most one of the two matchings. Constraints (13) express the subtour elimination constraints, and constraints (14) and (15) are the integrality constraints. Observe that this formulation not only has an exponential number of constraints, it also has an exponential number of variables. We describe in Section 4 how we deal with this feature when solving this formulation.

When we speak of the solution of the linear relaxation of FBP (which we denote by LFBP), we refer to a solution (z_M^b, z_M^r) satisfying (10)-(13), $z_M^b, z_M^r \ge 0$, for which the value $|\sum_{M\in\mathcal{M}_B} c^b(M)z_M^b - \sum_{M\in\mathcal{M}_R} c^r(M)z_M^r|$ (which we denote by v_{BP}) is minimum.

3.3 Comparing formulations FBB and FBP

Let us compare the linear relaxations of formulations FBB and FBP. Recall that, when given an instance I of ETSP, $v_{BB}(I)$ ($v_{BP}(I)$) denotes the value of LFBB (LFBP) when applied to instance I. Following standard terminology (see Vielma (2015) and references contained therein), we say that the linear relaxation of FBP is *stronger* than the relaxation of FBB when the two following conditions are fulfilled:

C1: for each instance I of ETSP, $v_{BP}(I) \ge v_{BB}(I)$,

C2: there exists an instance I of ETSP for which $v_{BP}(I) > v_{BB}(I)$.

Theorem 4. *The linear relaxation of FBP (i.e., LFBP) is stronger than the linear relaxation of FBB (i.e., LFBB).*

Proof. First, to prove C1, we show that any feasible solution to LFBP corresponding to some instance I can be transformed into a feasible solution of LFBB, while the costs of these two solutions are equal.

Consider a feasible solution (z_M^b, z_M^r) of LFBP. Construct a solution to LFBB as follows:

$$\text{for each edge } e \in E_B, x_e^b := \sum_{M\in\mathcal{M}_B: e\in M} z_M^b, \tag{16}$$

$$\text{for each edge } e \in E_R, x_e^r := \sum_{M\in\mathcal{M}_R: e\in M} z_M^r. \tag{17}$$

We need to show that (x_e^b, x_e^r) satisfies constraints (3-6), and that the solutions to LFBB and LFBP have an equal objective value. Let us consider constraints (3). Since all matchings are perfect matchings, it follows that each matching $M \in \mathcal{M}_B$ includes a single edge incident to each v. Consider some node $v \in V$. We have:

$$\sum_{e \in \delta(v) \cap E_B} x_e^b = \sum_{e \in \delta(v) \cap E_B} \sum_{M \in \mathcal{M}_B : e \in M} z_M^b = \sum_{M \in \mathcal{M}_B} z_M^b = 1 \tag{18}$$

The first equality follows from (16), and the second equality follows from the fact that the matchings in $\mathcal{M}_B$ that contain an edge e incident to node v are pairwise distinct matchings (since, by definition, no perfect matching can have two edges incident to node v). Finally, the last equality follows from (10). Thus, we conclude from the validity of (18) that constraint (3) is satisfied, and by a similar argument so is constraint (4).

Next, it is easily checked that constraint (5) is satisfied by (x_e^b, x_e^r), since constraint (12) is satisfied by (z_M^b, z_M^r); indeed, their left hand sides are equal by construction (16-17).

We now turn to the subtour elimination constraints. Observe that for each $S \subset V$, with $|S|$ even, and $4 \leq |S| \leq |V| - 4$:

$$\sum_{e \in \delta(S) \cap E_B} x_e^b = \sum_{e \in \delta(S)} \sum_{M \in \mathcal{M}_B :\ e \in M} z_M^b \text{ and } \sum_{e \in \delta(S) \cap E_R} x_e^r = \sum_{e \in \delta(S)} \sum_{M \in \mathcal{M}_R : e \in M} z_M^r. \tag{19}$$

It follows that

$$\sum_{e \in \delta(S) \cap E_B} x_e^b + \sum_{e \in \delta(S) \cap E_R} x_e^r = \sum_{e \in \delta(S)} \Big(\sum_{M \in \mathcal{M}_B : e \in M} z_M^b + \sum_{M \in \mathcal{M}_R : e \in M} z_M^r \Big) \geq 2.$$

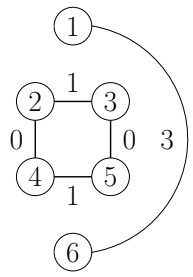
Thus, when given a feasible solution (z_M^b, z_M^r), the solution (x_e^b, x_e^r) constructed using (16)-(17) is a feasible solution to LFBB.

It remains to show that both solutions have an equal objective value. This is true by construction:

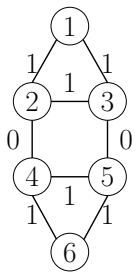
$$\sum_{e \in E_B} d_e^b x_e^b = \sum_{e \in E_B} d_e^b \sum_{M \in \mathcal{M}_B : e \in M} z_M^b = \sum_{M \in \mathcal{M}_B} \Big(\sum_{e \in E_B} d_e^b \Big) z_M^b = \sum_{M \in \mathcal{M}_B} c^b(M) z_M^b.$$

We have now shown that if there exists a feasible solution (z_M^b, z_M^r) to LFBP, we can construct a feasible solution (x_e^b, x_e^r) with an equal value. It follows that, for each instance I, we have $v_{BP}(I) \geq v_{BB}(I)$.

Let us now turn to condition C2. We exhibit an instance I of ETSP, for which $v_{BP}(I) > v_{BB}(I)$. Let $|V| = 6$, $E_B = \{(v_1, v_6), (v_2, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5)\}$ and $E_R = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5), (v_4, v_6), (v_5, v_6)\}$. We refer to Figure 2 for an illustration of the instance where the edge-costs are depicted near the corresponding edges.



(a) E_B



(b) E_R

Figure 2: Edge Sets

It can be checked $x_{16}^b = x_{24}^b = x_{35}^b = 1$ and $x_{12}^r = x_{13}^r = x_{23}^r = x_{45}^r = x_{46}^r = x_{56}^r = 0.5$ constitutes a feasible solution for LFBB, with $v_{BB}(I) = 0$ (see Figure 3 showing this solution).

It is also true that there does not exist a solution feasible to LFBP with $v_{BP}(I) = 0$. To see this, notice that both $\mathcal{M}_B$ and $\mathcal{M}_R$ contain only two perfect matchings. Both matchings in $\mathcal{M}_B$ contain the edge (v_1, v_6). Thus, for each $M \in \mathcal{M}_B$, we have $c^b(M) \geq 3$ and $\sum_{M \in \mathcal{M}_B} c^b(M) z_M^b \geq 3$. For $\mathcal{M}_R$, both matchings have two edges with weight 1 and one edge with weight 0. Summing these, we find $\sum_{M \in \mathcal{M}_R} c^r(M) z_M^r = 2$. Thus, for each solution to this instance of LFBP, we have $v_{BP}(I) \geq 1$.

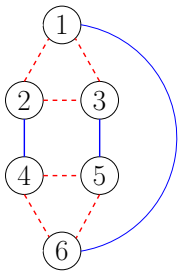


Figure 3: Solution to LFBB

We have now shown that for each instance of ETSP, the value of an optimal solution of LFBB is at least as low as the value of an optimal solution to LFBP and that there exist instances for which the value of an optimal solution of LFBB is lower than for LFBP. We conclude that formulation FBP is stronger than formulation FBB. □

Although Theorem 4 suggests to prefer Formulation FBP over FBB, there is a relevant set of instances for which the value of the linear programming relaxations of the two models coincide:

Theorem 5. *If $E_B = E_R$ and if $d_e^b = d_e^r$ for each $e \in E$, then $v \equiv v_{BP}(I) = v_{BB}(I)$, with $v \in \{0, \infty\}$ for each instance I.*

Proof. Let I be an instance of ETSP, with $E_B = E_R$ and $d_e^b = d_e^r$ for each $e \in E$, for which there exists a feasible solution (x_e^b, x_e^r) to LFBB. Consider now the following solution $(\bar{x}_e^b, \bar{x}_e^r)$, with $\bar{x}_e^b = \bar{x}_e^r = (x_e^b + x_e^r)/2$. Observe that $(\bar{x}_e^b, \bar{x}_e^r)$ is both feasible (as $E_B = E_R$) and has value 0 (as $d_e^b = d_e^r$). We now claim that there exists a feasible solution to LFBP with value 0, as both $\bar{x}_e^b$ and $\bar{x}_e^r$ lie within the perfect matching polytope. Indeed, consider the following set of linear inequalities describing the perfect matching polytope (Edmonds (1965); Schrijver (2003)).

$$x_e \geq 0 \qquad \forall e \in E. \tag{20}$$

$$\sum_{e \in \delta(v) \cap E} x_e = 1 \qquad \forall v \in V \tag{21}$$

$$\sum_{e \in \delta(S) \cap E} x_e \geq 1 \qquad \forall S \subset V \text{ with } |S| \text{ odd }. \tag{22}$$

Inequalities (20) and inequalities (21) are obviously satisfied. Indeed any feasible solution to LFBB must satisfy these, as they are equivalent to respectively (the linear relaxation of) constraints (7) and (4) for blue, and constraints (8) and (5) for red. Furthermore, since $E_b = E_r$, and $\bar{x}_e^b = \bar{x}_e^r$ for each edge, constraints (6) imply that

$$\sum_{e \in \delta(S) \cap E} 2\bar{x}_e^b \geq 2 \qquad \forall |S| \geq 3,$$

$$\sum_{e \in \delta(S) \cap E} 2\bar{x}_e^r \geq 2 \qquad \forall |S| \geq 3.$$

As a consequence, inequalities (22) are also satisfied. Since both $\bar{x}_e^b$ and $\bar{x}_e^r$ lie within the perfect matching polytope, they can both be described by a convex combination of perfect matchings (Edmonds (1965); Schrijver (2003)), represented by z_M^b, z_M^r respectively. These convex combinations form a solution to LFBP, which is equivalent to (x_e^b, x_e^r).

To conclude, we have shown (in the proof of Theorem 4) that the existence of a feasible solution to LFBP implies the existence of a feasible solution to LFBB with the same objective value. We have now shown that if $E_B = E_R$ and if $d_e^b = d_e^r$ for each $e \in E$, a feasible solution to LFBB implies the existence of a feasible solution with value 0 for both LFBB and LFBP. Thus, when $E_B = E_R$ and $d_e^b = d_e^r$ for all $e \in E$, there are only two possibilities: either both linear relaxations have a solution with value 0, or both linear relaxations are infeasible. □

Notice that there are two assumptions in Theorem 5. Each of these assumptions is necessary to achieve equality of $v_{BP}(I)$ and $v_{BB}(I)$: the example in the proof of Theorem 4 shows that $E_B = E_R$ is needed, and the instance described below implies that $d_e^b = d_e^r$ for each e is needed as well. For example, consider the following instance. Let

$$|V| = 6 \text{ and } E_b = E_r = \{(v_1, v_2), (v_1, v_3), (v_1, v_6), (v_2, v_3), (v_2, v_4), (v_3, v_5), (v_4, v_5), (v_4, v_6), (v_5, v_6)\}.$$

The weight of all edges, for both red and blue is one, except for $d_{16}^r = d_{24}^r = d_{35}^r = 0$. It can be checked that a solution exists with $v_{BB}(I) = 0$. For LFBP, each blue matching has weight 3, and each red matching has a weight ≤ 2, thus $v_{BP}(I) > 0$.

Furthermore, we point out that, in case the instance I admits a feasible solution to ETSP, and assuming that $E_B = E_R$ and if $d_e^b = d_e^r$ for each e, the two values $v_{BP}(I)$ and $v_{BP}(I)$ not only coincide as predicted by Theorem 5, but are in fact equal to 0.

Finally, we note that the proof of Theorem 5 suggest a way of strengthening formulation FBB in such a way that it becomes equivalent to FBP. Imposing conditions (22) on both x_e^b and x_e^r implies all solutions can be described as convex combinations of perfect matchings, and thus that for any solution there exists a solution to LFBP with equal objective value.

4 A branch-and-price-and-cut approach for solving model FBP

In this section, we describe how we solve model FBP using a branch-and-price-and-cut approach. For a general description of this methodology, we refer to Desrosiers and Lübbecke (2011). There are a number of key ingredients in this approach: adding cuts (Section 4.1), solving the pricing problem (Section 4.2) and what branching rule is used (Section 4.3). An overall view of our procedure is given in Figure 4; we now give a detailed description of the key ingredients.

4.1 Adding Cuts

The presence of an exponentially large number of constraints (the subtour elimination constraints (13)) in model FBP gives rise to a challenge. Indeed, when solving the linear relaxation of FBP, it is not a good idea to explicitly add all subtour elimination constraints, as there are simply too many of them. Instead, we opt to add only those inequalities that are violated by a feasible solution found at some stage of the procedure. This allows us to keep the number of constraints used in the model, limited. More specifically, we start out by solving a model that (i) uses only a subset of the possible variables (see Section 6.1.2 for details), and (ii) contains only constraints (10), (11) and (12). Next, we find out whether a violated inequality of the form (13) exists, i.e., we separate over (13). This is done as follows: first, we translate the solution (z_M^b, z_M^r) to an x-solution using (16)-(17). Then we use a min-cut routine to establish whether a violated solution exists. If so, the corresponding inequality is added to the formulation, and we resolve the model. This process repeats itself until the solution satisfies (13). Notice that in each iteration we add at most a single violated inequality.

4.2 Pricing Problem

The pricing problem for the linear programming relaxation of formulation FBP amounts to establishing whether there exists a variable z_M^b (or z_M^r) with negative reduced costs. When we associate dual variables v, w, u_e and a_S to constraints (10), (11), (12), and (13) respectively, linear programming theory tells us that the reduced costs of variable z_M^b ($M \in \mathcal{M}_B$) equal:

$$c_M^b - v - \sum_{e \in M} u_e - \sum_{S \subset V, |S| \geq 3} |\delta(S) \cap M| a_S. \tag{23}$$

A similar expression can be written down for the reduced costs of variable z_M^r.

We claim that, given the dual variables, the existence of a variable with negative reduced costs can be detected by computing a minimum-weight perfect matching problem.

Lemma 6. *The pricing problem corresponding to formulation LFBP can be solved by computing a minimum-weight perfect matching.*

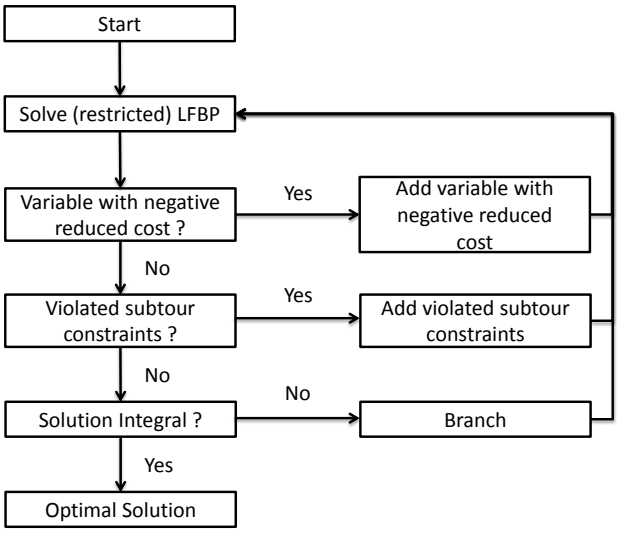


Figure 4: A schematic overview of the branch-and-price-and-cut algorithm.

Proof. Consider the graph (V, E_B), and for each edge $e \in E_B$, introduce edge costs γ_e defined as follows:

$$\gamma_e = d_e^b - u_e - \sum_{S \subset V : e \in \delta(S), |S| \geq 3} a_S. \tag{24}$$

Notice that for an edge $e \in E_B \setminus E_R$, no u_e exists; in that case, this term disappears from the expression above. Suppose that we have a minimum weight perfect matching M^* in (V, E_1) with respect to the edge costs γ_e. We have, using (24):

$$\begin{aligned} \sum_{e \in M^*} \gamma_e &= \sum_{e \in M^*} (d_e^b - u_e - \sum_{S \subset V : e \in \delta(S), |S| \geq 3} a_S) \\ &= c^b(M^*) - \sum_{e \in M^*} u_e - \sum_{S \subset V : e \in \delta(S) \cap M^*, |S| \geq 3} a_S. \end{aligned}$$

Thus, if $\sum_{e \in M^*} \gamma_e < v$, it follows that $c^b(M^*) - \sum_{e \in M^*} u_e - \sum_{S \subset V : e \in \delta(S) \cap M^*, |S| \geq 3} a_S < v$ and, hence (23) implies that there is a variable with negative reduced costs. In addition, if $\sum_{e \in M^*} \gamma_e \geq v$, it is a fact that no variable z_M^b with negative reduced costs exists. □

We solve a pricing problem for each of the two edge sets. Strictly speaking it suffices to halt the pricing problem after a variable z_M^b with negative reduced costs is discovered; in practice, however, better results are obtained when multiple columns are returned simultaneously. Thus, in a single iteration, we add at most two variables to the model, at most one per color.

4.3 Branching

After solving the linear relaxation of FBP to optimality, the resulting solution may be fractional, that is, there may exist matching(s) $M \in \mathcal{M}_B$ with $0 < z_M^b < 1$ or matching(s) $M \in \mathcal{M}_R$ with $0 < z_M^r < 1$. Hence a branching rule is required. If there is a fractional solution, then there is an edge $e^* = (i, j)$ such that either $0 < \sum_{M \in \mathcal{M}_B :\ e^* \in M} z_M^b < 1$ or $0 < \sum_{M \in \mathcal{M}_R :\ e^* \in M} z_M^r < 1$ holds, or both. Indeed, one easily verifies that integrality of the z_M^b, z_M^r variables implies that each edge is either part of the blue matching, or part of the red matching, or not used at all. This allows us to specify the following branching rule. Suppose that, for some edge $e^* \in E_B$: $0 < \sum_{M \in \mathcal{M}_B :\ e^* \in M} z_M^b < 1$ holds. There exist two possibilities: either, in an incumbent solution edge $e^* = (i, j)$ is part of the blue matching, or it is not. In the former case all edges in E_B that are incident to vertices i or j (except edge $e^* = (i, j)$) are removed from E_B,

and edge (i, j) (if present in E_R) is removed from set E_R. In the latter case, edge e^* is removed from set E_B. The case where $0 < \sum_{M \in \mathcal{M}_R:\ e^* \in M} z^r_M < 1$ works completely analogously.

This branching rule has the property that it solely affects the edge sets E_B and E_R. Thus, each node in the search tree corresponds to an instance solely defined by a specific graph. In our implementation, we select edge e^* based on the proximity to 0.5: the edge for which the value $\sum_{M \in \mathcal{M}_B:\ e^* \in M} z^b_M$ is closest to 0.5 is selected for branching.

5 Local Search

In order to asses the potential of local search based methods, we investigate the performance of a basic greedy local search algorithm for ETSP. We use the outcomes of this algorithm to better understand the difficulty of different classes of instances. Moreover, we also use the value of a solution (i.e., the value of this local optimum) found by the local search algorithm as an upper bound when executing the exact methods.

The idea of the local search algorithm is based on the well-known 2-opt neighbourhood for the TSP, which goes back to Croes (1958). Since, in ETSP, the colour of the edges must alternate, we distinguish two versions of this neighbourhood which depend on whether the two removed edges have the same colour, or not. Consider a feasible tour containing blue edges $(1, 2), (3, 4), \ldots, (n-1, n)$ and red edges $(2, 3), (4, 5), \ldots, (n, 1)$. In mono-colour 2-opt, we remove, given a feasible solution the ETSP, two identically coloured edges from the tour, say (i, j) and (k, l). A new tour is then formed by adding two edges of the same colour, say (i, k) and (j, l).

In bi-colour 2-Opt, we remove two distinctly coloured edges from the tour. Observe that simply reconnecting the edges as in mono-colour 2-Opt is not possible, as no combination of red and blue colourings of the new edges will satisfy the alternating colour property. Consider a move where one blue edge (c, d) and one red edge (k, l) are removed from the tour. The remaining edges form paths from d to k and from l to c. There are two ways to reconnect the tour, either by adding a blue edge (c, k) and a red edge (d, l) or a red edge (c, k) and a blue edge (d, l). In the first case, the colour of all edges $(d, e), \ldots, (j, k)$ must be switched, in the second case this is true for all edges $(l, m), \ldots, (b, c)$.

We use these neighbourhoods in a simple greedy local search algorithm. For a given solution, these neighbourhoods are searched until a better solution is found. This better solution then becomes the incumbent solution and in turn, its neighbourhoods are searched for a better solution.

It is natural to investigate a generalization of mono-colour 2-Opt in the following sense: while fixing the blue matching, consider all red matchings that, together with the blue matching, yield a feasible solution to ETSP. This is a larger neighbourhood than mono-colour 2-opt, and hence potentially more interesting. Unfortunately, deciding whether a given matching can be extended to a tour is NP-complete, see Bienkowski and Zalewski (2013). It follows that finding the best matching in this neighbourhood is hard, and hence, searching through this neighbourhood does not seem to be an attractive option.

6 Computational Experiments

In this section we first give (in Section 6.1) some details concerning the implementation of the two integer programming models, in particular model FBP. Then, in Section 6.2, we describe the instances that we use for our experiments whose outcomes are reported in Section 6.3.

6.1 Implementation Details

Our models do not presume the existence of a feasible solution. In fact, we find that "interesting" instances are those that are in some sense close to the threshold between feasibility and infeasibility (See Section 6.3). However, it is clear that the existence of a perfect matching in (V, E_B) as well as the existence of a perfect matching in (V, E_R) are necessary conditions for the existence of a feasible solution. We do assume that each instance satisfies these necessary conditions. Moreover, we use this assumption to establish an upper bound W on the value of a feasible solution (if one exists); and if no (heuristically

obtained) feasible solution is available, we use this upper bound at the root node of the search tree. The upper bound W is computed as follows:

$$W = \max\{\max_{M \in \mathcal{M}_B} c^b(M) - \min_{M \in \mathcal{M}_R} c^r(M), \max_{M \in \mathcal{M}_R} c^r(M) - \min_{M \in \mathcal{M}_B} c^b(M)\} + 1 \tag{25}$$

This bound is easy to calculate by computing both a min cost and a max cost perfect matching on graphs $G(V, E_B)$ and $G(V, E_R)$.

All experiments involving models FBB and FBP are conducted on an Intel Core i7-4790 (3.6GHz) machine. ILOG Cplex version 12.6.2 has been used to solve the Linear and Integer Programming problems. For Cplex, we used the default settings and a maximum of 2 solver threads. The Branch-and-Price approach is implemented through the Branch-and-Price framework provided by the Java OR Library [JORLIB] version 1.1. Again, we used the default settings and 2 solver threads. The pricing problem of the BPC approach (min cost perfect matching problem) has been solved using the implementation in the COIN-OR LEMON library [LEMON]. Separation of subtours in both LFBB and LFBP is implemented through the min-cut routine in JgraphT version 0.9.1 [JgraphT].

6.1.1 Model FBB

We solve model FBB by Cplex using default settings. In particular, consider the search tree induced by solving model (FBB) without constraints (6). A node in this tree corresponds to solving a linear program. The solution of such a node may be fractional, or integral. If the solution of a node is fractional, i.e., if there exists an edge $e \in E_b \cup E_r$ such that $0 < x_e^b < 1$ or $0 < x_e^r < 1$, a standard branching rule is used to create two nodes. If however, the solution is integral, we separate over constraints (6). Then, if a violated inequality is found, we add it to the current model, and resolve it; if no violated inequality is found, we have found a feasible solution to our problem: we update the incumbent, and close this node.

Of course, an alternative implementation would be separate over constraints (6) irrespective of the integrality of the solution found, i.e., in each node of the search tree. Preliminary experiments suggest that this approach gives rise to longer running times, hence in our experiments we adopted the approach described above.

6.1.2 Model FBP

The implementation of the FBP framework is fully deterministic and relies on reversible data structures. Instead of copying entire MIP models, graph structures, etc, each branch in the search tree introduces a number of reversible changes to the underlying data structures. When backtracking in the search tree, changes are automatically reverted. Since changes can be made locally, this is much cheaper and faster than copying and modifying entire data structures, especially if only small changes need to be made. Furthermore, this makes it much easier to implement different search strategies based on for example Depth First Search, Breadth First Search or Strong Branching techniques. In this work we use Depth First Search to minimize the overhead involved when backtracking.

Initialization Each node of the search tree must be initialized with a set of variables allowing a feasible solution to the linear relaxation of FBP, or one must show that, given the edge sets E_b and E_r no feasible solution exists, rendering the node infeasible.

Due to the presence of the subtour elimination constraints (13), and due to the fact that the graphs may be incomplete, it may be difficult to generate a set of variables (matchings) that admit a feasible solution satisfying constraints (10)-(13). And even if this would be possible, then this would require to solve a separate subproblem at each node of the search tree to establish a feasible initial solution. We avoid these issues by solving the linear relaxation of a slightly more general formulation of model FBP.

Indeed, by introducing a slack variable λ in the constraints (10), (11), (13), a model is obtained for

which a trivial initial solution exists. Model FBP' is defined as follows:

$$FBP' : \min \quad \Big|\sum_{M\in\mathcal{M}_B} c^b(M)z^b_M - \sum_{M\in\mathcal{M}_R} c^r(M)z^r_M\Big| + \lambda U \tag{26}$$

$$\text{s.t.} \quad \sum_{M\in\mathcal{M}_B} z^b_M + \lambda = 1 \tag{27}$$

$$\sum_{M\in\mathcal{M}_R} z^r_M + \lambda = 1 \tag{28}$$

$$\sum_{M\in\mathcal{M}_B:\ e\in M} z^b_M + \sum_{M\in\mathcal{M}_R:\ e\in M} z^r_M \le 1 \qquad \forall e \in E_B \cap E_R \tag{29}$$

$$\sum_{e\in\delta(S)} \Big(\sum_{M\in\mathcal{M}_B:e\in M} z^b_M + \sum_{M\in\mathcal{M}_R:e\in M} z^r_M\Big) + 2\lambda \ge 2 \qquad \forall S \subset V, |S| \ge 3 \tag{30}$$

$$z^b_M \in \{0,1\} \qquad \forall M \in \mathcal{M}_B \tag{31}$$

$$z^r_M \in \{0,1\} \qquad \forall M \in \mathcal{M}_R \tag{32}$$

In this model, original constraints (10), (11), (13) may be violated, but any violation is penalized in the objective, where $\lambda U > 0$ is the penalty incurred when $\lambda > 0$, with U being a fixed constant.

From the construction of model FBP' it is apparent that there is always a feasible solution having objective value U: simply set $z^r_M = z^b_M = 0$ for all $M \in \mathcal{M}_1 \cup \mathcal{M}_2$, and $\lambda = 1$. Furthermore, it is clear that the pricing problem can be solved through the implementation sketched in Section 4.2. Finally, observe that by setting $\lambda = 0$, FBP' becomes identical to the linear relaxation of FBP. In fact, FBP' only yields a feasible solution to the linear relaxation of FBP iff $\lambda = 0$. To prove infeasibility of FBP', we must guarantee that any solution in FBP' with $\lambda > 0$ is more expensive in terms of the objective value than any solution to FBP' with $\lambda = 0$. This can be achieved by setting U to a large value. However, it is well known that this reduces the stability of the column generation procedure and potentially introduces numerical issues while implementing the model. Recall that a node in the BPC tree can be pruned if either of the following conditions holds:

1. the node is infeasible
2. the lower bound on the node exceeds the upper bound, i.e., the best incumbent integer solution.

Hence it suffices to ensure that there does not exist a solution to FBP' with $\lambda > 0$ having an objective value less than W, where W is the value of the best incumbent integer solution or any other valid upper bound on the optimal objective value (see (25)). The latter can be achieved by the following procedure:

Algorithm 1: Penalty update procedure for FBP'

```
U=W;
repeat
    solve FBP'. Let ω be the resulting objective value;
    if λ = 0 then
        Feasible solution for the linear relaxation of FBP has been found with objective value ω;
    else                                                        /* 0 < λ ≤ 1 */
        U := U/λ ;                                              /* Increase penalty */
until λ = 0 ∨ ω ≥ W + 1;
```

When the procedure terminates, either a solution to FBP' with $\lambda = 0$ is discovered, or a solution with $\lambda > 0$, $\omega \ge W$ is obtained in which case the node can be pruned.

6.1.3 Local Search

The local search experiments are conducted on an Intel Core i5-2520 (2.5GHz) machine using custom code. For each instance, the local search is run using 10 random tours as a starting point. The local search algorithm works on instances where the graph is complete, and where there is a blue cost d^b_e and a red cost d^r_e for each edge $e \in E$. Given an instance of ETSP for which either the blue edgeset E_B or the red edgeset E_R is not complete, we use the following procedure to construct an instance satisfying the required properties. Let H be the weight of the heaviest blue or red edge in absolute value ($H = \max_{e\in E}\{|d^b_e|, |d^r_e|\}$).

$$\text{For each } e \notin E_b, \text{ set } d^b_e = nH.$$
$$\text{For each } e \notin E_r, \text{ set } d^r_e = -nH.$$

Name of instance	obj	$t(s)$	# Optimal Random Tours
burma 14	0	0.00	3
ulysses 16	0	0.00	1
ulysses 22	0	0.00	2
berlin 52	0	0.01	1
eil 76	0	0.00	35
gr 96	0	0.02	0
gil 262	0	0.01	6
pr 264	0	0.03	1
lin 318	0	0.03	2

Table 1: Results for instances of Type 1

This choice of costs ensures that if the local search algorithm terminates with a solution whose value is at least as large as nH, no feasible solution to the original instance was found.

6.2 Instances

In the computational experiments, we use four sets of instances, named instances of Type 1, Type 2, Type 3, and Type 4. The instances of Types 1, 2, and 3 are based on TSPLib instances. Recall that the number of vertices in an instance of the ETSP needs to be even. If this is not the case for a particular TSPLib instance, we remove the last vertex.

For instances of Type 1, we use the nine smallest instances from TSPLib having between 14 and 318 nodes. Since the instances from TSPLib are geometric, the instances feature a complete graph with $E \equiv E_B = E_R$, and $d_e^b = d_e^r$ for each $e \in E$.

In order to study the behavior of our models on more interesting instances, we adapt the instances of Type 1 by removing edges. More in particular, for an instance of Type 1, we remove a certain number of the most expensive edges, thereby obtaining an incomplete graph $G(V, E)$. An edge (i, j) can only be removed if the remaining graph remains Hamiltonian; iteratively removing a single edge from the graph produces a new instance in each iteration. We only include instances (i) with a non-zero optimum value, and (ii) for which the optimal solution changed when compared to the solution found in the previous iteration. In total, this procedure gives us 29 instances of Type 2. Recall that these instances still have $E_B = E_R$, and $d_e^b = d_e^r$ for each $e \in E$.

Instances of Type 3 are constructed using the asymmetric TSP instances from TSPLib. The original TSPLib instances are defined on complete graphs. To construct our instances, we give each edge a 20% chance of only being in E_B, a 20% chance of only being in E_R, and a 10% chance of being a member of both. Furthermore, for an edge (i, j) we set $d_{(i,j)}^b = d(i, j)$ and $d_{(i,j)}^r = d(j, i)$. In this way, we construct 19 instances for which $E_B \neq E_R$, and for which there exist e with $d_e^b \neq d_e^r$.

Finally, instances of Type 4 are generated using complete graphs, with $E = E_B = E_R$. For each edge $e \in E$, the blue costs d_e^b are randomly generated from a uniform distribution between 201 and 300, while the red costs d_e^r are randomly generated from a uniform distribution between 1 and 100. As such, these instances have the property that for each edge $e \in E$, $d_e^b \neq d_e^r$ and the value of an optimum solution (as well as the value of the linear relaxations) can not be equal to 0. We use $|V| = 40, 50, 70, 90, 110$, and we generate 10 graphs for each value of $|V|$, leading to 50 instances in total.

6.3 Experimental Results

Let us first consider the instances of Type 1. The results of the local search algorithm are given in Table 1, where the column called 'obj' denotes the value of the solution found, where the column '$t(s)$' stands for the computation time needed (in seconds), and where the final column gives the number of optimum solutions among a set of 10.000 randomly generated tours.

From Table 1 it follows that instances of Type 1 are not difficult to solve. Indeed, the (simple) local search algorithm finds an optimum solution for each instance in negligible computing times. Perhaps even more telling, simply randomly generating a moderate number of tours will produce an optimum solution. Therefore, we chose not to run the exact approaches on instances of Type 1.

Consider now the instances of Type 2. Outcomes of the local search algorithm, model FBB, and model FBP are reported in Table 2, where the second column stands for the number of edges present in the instance, where the column called 'OPT' stands for the optimum value, where a column called 'nodes'

stands for the number of nodes in the search tree corresponding to that instance, and where the column called 't_{master}' ('$t_{pricing}$') stands for the time spent solving the master problem (the pricing problem).

From this table, we conclude that the local search algorithm is no longer effective: for the majority of instances it fails to find even a feasible solution (while there exists one). The two exact approaches are very efficient on these instances, and find an optimum solution usually within a second, and always within four seconds. On most instances, model FBB is a bit faster, while the number of nodes in the search tree of model FBP is usually less than the corresponding number for model FBB. Notice that Theorem 4 tells us that the values of the linear programming relaxations of these models equal 0 for instances of Type 2.

Table 2: Results for instances of Type 2

Name of	Number of		Local Search	Model FBB			Model FBP				
instance	Remaining Edges	OPT	obj	$t(s)$	$t(s)$	nodes	$t(s)$	t_{master}	$t_{pricing}$	nodes	cols
burma14_r50	40	1	45	0.00	0.16	389	1.04	1.18	1.14	198	344
burma14_r54	37	22	70	0.01	0.12	52	0.30	0.20	0.26	32	52
burma14_r55	36	35	INF	0.00	0.06	15	0.11	0.08	0.08	11	17
burma14_r56	35	97	INF	0.00	0.04	0	0.07	0.05	0.03	5	9
burma14_r57	34	234	INF	0.01	0.08	0	0.04	0.03	0.04	2	3
burma14_r63	31	258	INF	0.00	0.06	0	0.03	0.02	0.04	2	2
burma14_r71	25	305	INF	0.00	0.04	0	0.02	0.01	0.01	3	2
ulysses16_r81	38	1	INF	0.00	0.16	490	1.12	0.91	1.22	180	241
ulysses16_r90	32	10	INF	0.00	0.19	52	0.24	0.12	0.15	19	26
ulysses16_r103	23	14	INF	0.00	0.07	0	0.02	0.01	0.02	1	2
ulysses22_r186	50	1	INF	0.00	0.12	181	0.61	0.63	0.92	72	87
ulysses22_r191	45	97	INF	0.00	0.13	40	0.04	0.02	0.09	1	2
ulysses22_r204	36	101	INF	0.00	0.08	0	0.08	0.08	0.06	2	2
ulysses22_r216	28	157	INF	0.00	0.03	0	0.02	0.01	0.01	2	2
berlin52_r1277	82	2	INF	0.05	0.30	155	3.71	1.52	2.15	92	94
berlin52_r1285	77	12	INF	0.04	0.28	3	0.23	0.09	0.09	3	4
berlin52_r1293	70	28	INF	0.04	0.25	7	0.17	0.08	0.07	2	2
berlin52_r1299	64	37	INF	0.03	0.17	0	0.21	0.06	0.07	1	2
eil76_r2744	119	3	INF	0.11	0.17	3	0.93	0.67	0.57	8	6
eil76_r2751	116	10	INF	0.13	0.29	4	0.68	0.53	0.44	12	6
gr96_r4415	168	29	INF	0.42	0.35	43	0.79	0.35	0.42	1	2
gr96_r4421	165	341	INF	0.38	0.63	24	0.85	0.33	0.71	2	2
gr96_r4439	156	387	INF	0.27	0.48	5	0.75	0.32	0.35	4	5
gr96_r4447	149	432	INF	0.41	0.39	5	1.50	0.66	0.21	8	12
gr96_r4456	146	688	INF	0.28	0.28	3	0.54	0.16	0.24	1	2
gr96_r4460	144	994	INF	0.45	0.58	7	0.65	0.24	0.36	1	2
gr96_r4479	130	1013	INF	0.43	0.34	3	0.51	0.21	0.16	2	2
gr96_r4495	123	1016	INF	0.41	0.23	0	0.42	0.07	0.11	2	2
gr96_r4518	113	1199	INF	0.38	0.18	0	0.18	0.04	0.07	2	2

Let us now turn to instances of Type 3, whose outcomes are reported in Table 3. First, the local search algorithm works reasonably well, finding an optimum solution for the majority of instances. Second, model FBB solves all instances much faster than model FBP. In fact, since we allotted each model 1800 seconds for each instance, there are three instances that model FBP cannot solve within this time-limit. Interestingly, the values of the linear programming relaxations are equal except for one particular instance (although Theorem 4 does not apply for instances of Type 3).

Table 3: Results for instances of Type 3

		Local Search		Model FBB			Model FBP					
Name	OPT	obj	$t(s)$	v_{BB}	$t(s)$	nodes	v_{BP}	$t(s)$	t_{master}	$t_{pricing}$	nodes	cols
br17	0	11	0	0	0.29	118	0	0.95	0.41	0.30	60	179
ftv33	0	3	0.01	0	0.64	795	0	8.07	5.38	1.38	481	1363
ftv35	0	0	0.02	0	0.39	91	0	4.26	2.74	0.80	179	403
ftv38	0	0	0.02	0	0.72	440	0	8.68	6.56	1.17	515	2552
p43	7611	7785	0.02	7606	0.38	0	7611	0.30	0.25	0.03	2	10
ftv44	0	0	0.02	0	1.17	1311	0	22.4	15.57	4.53	761	4929
ftv47	0	1	0.03	0	1.44	648	0	30.3	21.74	5.54	987	5326
ry48p	0	1	0.03	0	1.13	1000	0	34.3	26.87	4.91	799	4613
ft53	0	1	0.04	0	2.60	6628	0	12.3	10.38	0.95	356	944
ftv55	0	0	0.04	0	1.59	818	0	21.1	16.03	3.47	514	3058
ftv64	0	0	0.07	0	1.32	164	0	35.3	28.03	5.47	600	3765
ft70	0	0	0.09	0	2.66	1470	0	93.8	72.95	13.63	1560	7824
ftv70	0	0	0.05	0	2.59	1140	0	77.0	61.31	12.64	789	8277
kro124p	0	0	0.31	0	7.92	6539	0	1091	857.5	211	2438	58272
ftv170	0	0	0.33	0	27.35	7213	0	1800	1653	108	3747	17376
rbg323	0	0	0.87	0	63.66	209	0	1167	1140	13.46	270	468
rbg358	0	0	1.22	0	148.36	1865	0	631	620	6.08	93	153
rbg403	0	0	1.41	0	91.48	707	0	1800	1773	16.35	209	359
rbg443	0	0	1.78	0	0.62	5	0	1800	1779	15.33	165	352

Finally, we look at instances of Type 4 in Table 4. Recall that each number in this table is an average over 10 instances. The simple local search method works reasonable well, finding a solution about 6% above the optimum value. The two exact approaches are quite efficient again, finding optimum solutions within 30 seconds; model FBB is a bit faster than model FBP. Notice that in most cases the Cplex implementation of model FBB solves the problem without any branching. The solutions to the LP relaxations of both models are already very close to the optimal solutions, with average gaps below 0.05 % for both formulations. For 32 out of 50 instances, the linear relaxation of the FBP formulation gives a higher solution than the linear relaxation of the FBB formulation.

Table 4: Results for instances of Type 4 (averages)

	Local Search		Model FBB		Model FBP				
Instance Size	$t(s)$	Opt. Gap	$t(s)$	nodes	$t(s)$	t_{master}	$t_{pricing}$	cols	nodes
40	0	0.07	3.80	0	5.92	3.97	1.64	16.1	3.1
50	0.00	0.06	6.35	0	8.68	6.52	1.74	17.1	3.4
70	0.01	0.06	10.38	0	19.86	15.72	3.41	26	5
90	0.03	0.06	18.60	0.6	30.92	25.31	4.77	32.8	5
110	0.05	0.05	23.80	0.6	27.46	22.80	4.15	20.4	3.5

When studying the joint performance of the methods we implemented, we conclude as follows:

- The simple local search algorithm works when the instances are dense enough. Indeed, instances of Type 2 that are on the border of feasibility and infeasibility are not handled well by the local search, whereas instances of the other types are solved to optimality (Type 1 and Type 3), or within a reasonable deviation (Type 4).
- it is apparent that model FBB is faster than model FBP on instances of Type 3; however, for instances of Types 2 and 4, the performance is comparable. In particular, the reported averages for instances of Type 4 hide significant variance in the computation times for model FBP: in 25 out of 50 instances model FBP is faster than model FBB.
- The lack of strong bounds severely hampers model FBP when solving instances of Type 3 (and to a lesser extent Type 2). At each non-leave node, a lower bound of zero is encountered. Only at the leaves, non-zero integer solutions are discovered. Consequently it is very difficult to guide the search or to cut of branches due to the weak lower bounds. Indeed, theoretically, the main advantage of model FBP is the stronger linear relaxation, and model FBB only obtains better results on instances where the linear relaxation for both formulations is 0. For instances where the linear relaxation does come into play, results for both approaches are comparable.
- The majority of time, when using model FBP, is spent on solving the master problems and separating the subtour inequalities.
- A branching decision is made as soon as the master problem reaches a feasible solution equal to the lower bound of the node in the search tree corresponding to model FBP. Typically, this often happens after only a few iterations. Hence, the number of iterations, as well as the number of generated columns per node in this search tree is very low.
- The penalty update scheme (Algorithm 1) works well: the number of penalty updates is significantly lower than the number of nodes in the search tree corresponding to model FBP, meaning that for the majority of nodes no updates are required. This approach is computationally much cheaper than using a dedicated method to generate a feasible initial solution at each node of the tree (or to prove that such a solution does not exist).

7 Conclusion

We have introduced and analyzed the Equitable Traveling Salesman Problem (ETSP). We have shown that the problem is NP-Hard, even when the graph is complete. Two integer programming models are presented for the ETSP, and we compare their linear programming relaxations. One model can be solved

through a traditional branch-and-bound-and-cut approach, whereas the other model is embedded in a branch-and-price-and-cut framework. The pricing problem in the branch-and-price approach amounts to finding a minimum weight matching. Computational experiments on adapted TSPLib instances show that the best results are obtained with the branch-and-bound-and-cut approach.

References

K. Aardal, R. E. Bixby, C. A. J. Hurkens, A. K. Lenstra, and J. W. Smeltink, "Market split and basis reduction: Towards a solution of the Cornuéjols-Dawande instances," *INFORMS Journal on Computing*, vol. 12, no. 3, pp. 192–202, 2000.

T. Bassetto and F. Mason, "Heuristic algorithms for the 2-period balanced traveling salesman problem in euclidean graphs," *European Journal of Operational Research*, vol. 208, no. 3, pp. 253 – 262, 2011.

K. H. Becker, "Twin-constrained hamiltonian paths on threshold graphs," Ph.D. dissertation, London School of Economics, 2010.

Š. Berežný and V. Lacko, "The color-balanced spanning tree problem," *Kybernetika*, vol. 41, no. 4, pp. 539–546, 2005.

M. Bienkowski and P. Zalewski, "(1, 2)-hamiltonian completion on a matching," *International Journal of Foundations of Computer Science*, vol. 24, no. 01, pp. 95–108, 2013.

P. M. Camerini, F. Maffioli, S. Martello, and P. Toth, "Most and least uniform spanning trees," *Discrete Applied Mathematics*, vol. 15, no. 23, pp. 181 – 197, 1986.

P. Cappanera and M. G. Scutellà, "Balanced paths in acyclic networks: Tractable cases and related approaches," *Netw.*, vol. 45, no. 2, pp. 104–111, Mar. 2005.

G. Cornuéjols and M. Dawande, *A Class of Hard Small 0-1 Programs*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, vol. 1412, pp. 284–293.

G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.

E. Delcour, "Two salesmen and a bike," Master's thesis, Faculty of Business and Economics, KU Leuven, 2012.

J. Desrosiers and M. E. Lübbecke, "Branch-price-and-cut algorithms," *Wiley encyclopedia of operations research and management science*, 2011.

J. Edmonds, "Maximum matching and a polyhedron with 0, l-vertices," *J. Res. Nat. Bur. Standards B*, vol. 69, no. 1965, pp. 125–130, 1965.

A. Ficker, F. C. Spieksma, and G. J. Woeginger, "Balanced vector optimization," *Research Report KUL*, 2016.

JgraphT, "JgraphT," http://jgrapht.org.

JORLIB, "Java OR Library v1.1," http://jkinable.github.io/jorlib/.

N. Katoh and K. Iwano, "Efficient algorithms for minimum range cut problems," *Networks*, vol. 24, no. 7, pp. 395–407, 1994.

J. Kinable, "Decomposition Approaches for Optimization Problems," Ph.D. dissertation, Science, Engineering and Technology Group, Campus Kulak Kortrijk, Computer Science, Campus Kulak Kortrijk, Faculty of Engineering Science, Dec. 2014.

J. Larusic and A. P. Punnen, "The balanced traveling salesmanproblem," *Computers and Operations Research*, vol. 38, no. 5, pp. 868–875, May 2011.

LEMON, "COIN-OR Lemon Graph Library v1.3.1," https://lemon.cs.elte.hu/trac/lemon.

S. Martello, W. Pulleyblank, P. Toth, and D. de Werra, "Balanced optimization problems," *Operations Research Letters*, vol. 3, no. 5, pp. 275 – 278, 1984.

A. Schrijver, *Combinatorial optimization: polyhedra and efficiency.* Springer Science & Business Media, 2003, vol. 24.

J. P. Vielma, "Mixed integer linear programming formulation techniques," *SIAM Review*, vol. 57, no. 1, pp. 3–57, 2015.

Z. Zeitlin, "Minimization of maximum absolute deviation in integers," *Discrete Applied Mathematics*, vol. 3, no. 3, pp. 203 – 220, 1981.

FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be